Tampereen ammattikorkeakoulu

# Sovellusten ohjelmointi ja käytettävyys

## Oppimispäiväkirja

Antti Venetjoki

**SISÄLLYS**

# 1 Viikkoharjoitukset 1

## 1.1 Android -ympäristön asennus ja Hello World

### 1.1.1 Android -ympäristön asennus

Asensin version Android Studiosta, joka sisälsi myös Android SDK ja tarvittavat työkalut. Asennusohjelma ohjasi automaattisesti SDK asennukseen. Huomasin, että prosessi asensi myös virtuaaliset Android-laitteet (AVD), joita käytetään sovellusten testaamiseen ilman fyysistä laitetta. Valitsin AVD Managerista laitteeksi Pixel 8 ja Android 12 -version, jossa on API-versio 31. Virtuaalikoneen käynnistämisessä huomasin, että laite voi olla hidas, mikä johtui osittain koneen resurssien rajoitteista.

### 1.1.2 Github-linkki

github.com

### 1.1.3 Todiste ohjelman ajosta

## 1.2 Jetpack Compose -tutustuminen

### 1.2.1 Github-linkki

github.com

### 1.2.2 Koodi ajettuna Android virtuaalikoneessa


## 1.3 Kotlin essentials – osa 1

github.com

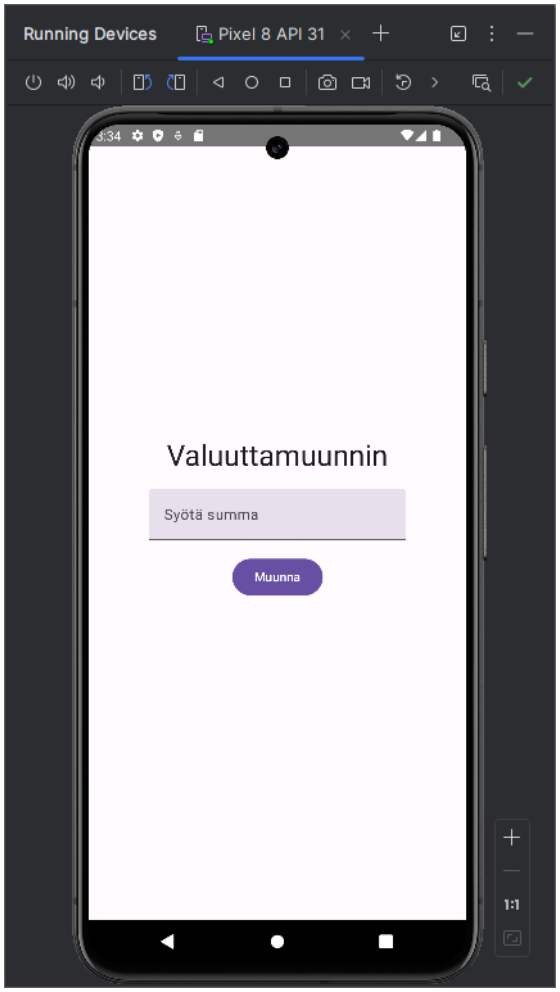
# 2 Viikkoharjoitukset 2

## 2.1 Valuuttamuuntimen käyttöliittymä

### 2.1.1 Github-linkki

github.com

### 2.1.2 Kuva käyttöliittymästä

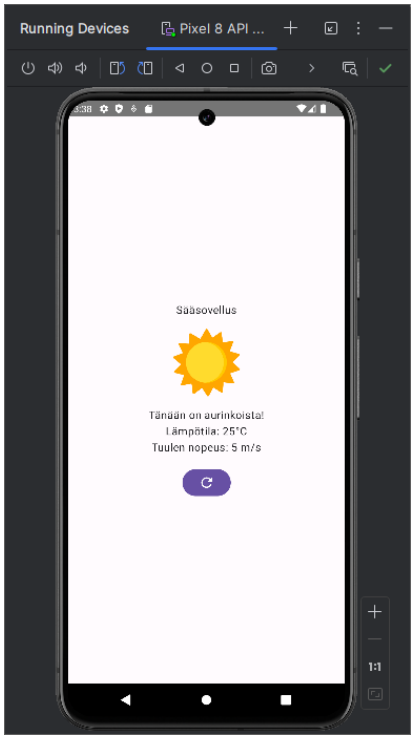



## 2.2 Sääsovelluksen käyttöliittymä

### 2.2.1 Github-linkki

github.com

### 2.2.2 Kuva käyttöliittymästä

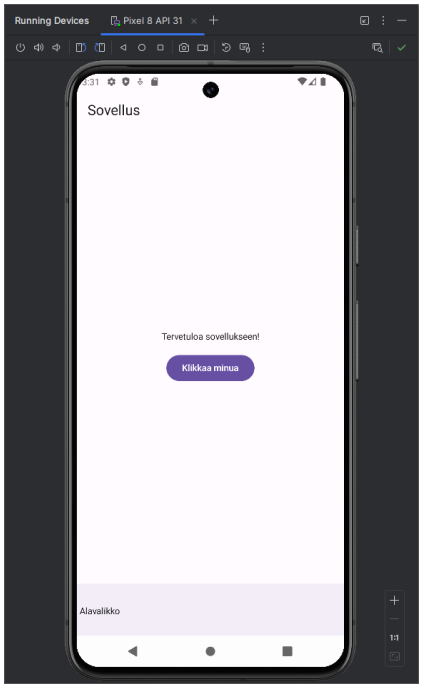



## 2.3 Scaffold

### 2.3.1 Github-linkki

github.com

### 2.3.2 Kuva käyttöliittymästä




## 2.4 Kotlin harjoituksia osa 2

github.com

# 3 Viikkoharjoitukset 3

## 3.1 Lokalisointi

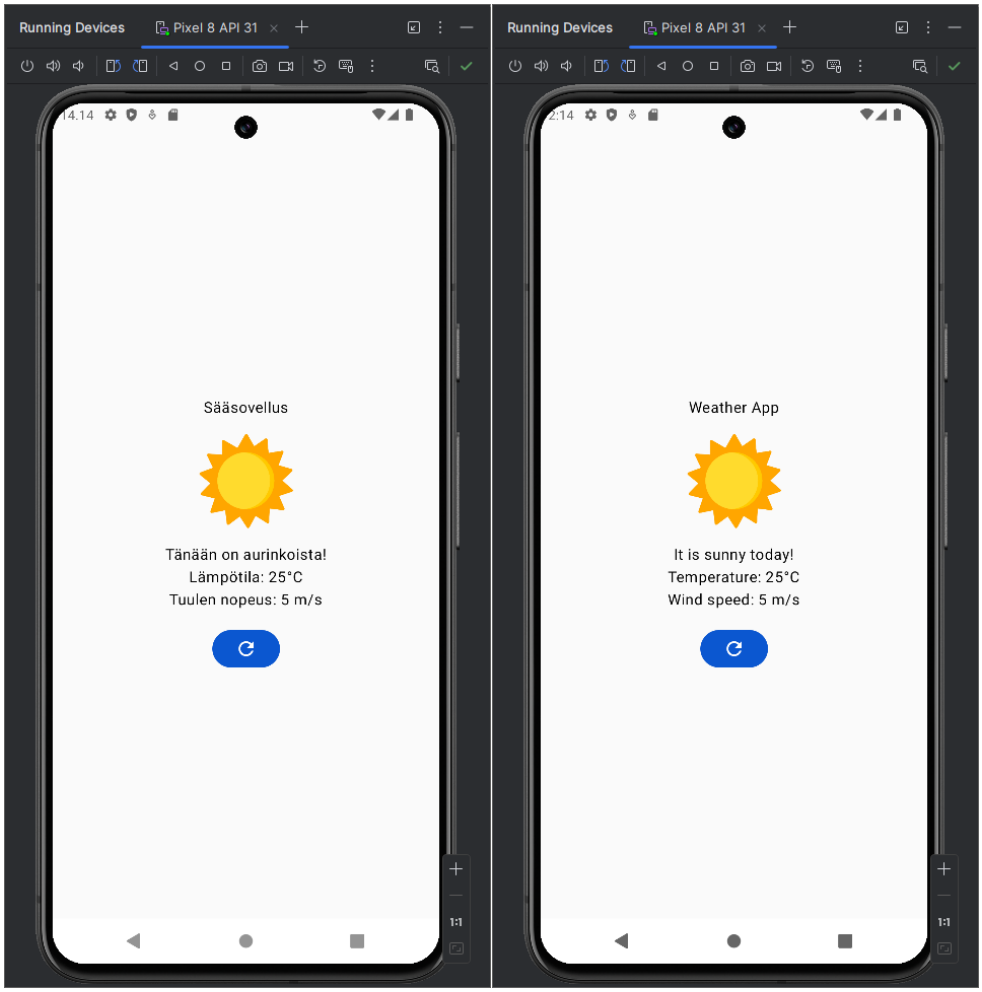
### 3.1.1 Pohdinta

Lokalisointi resurssitiedostoissa koodin sijaan on hyvä käytäntö monista syistä. Koodissa ei ole kovakoodattuja merkkijonoja, mikä tekee koodista siistimpää ja helpommin luettavaa. Resursseja voidaan muokata ilman, että kosketaan itse logiikkaan, mikä vähentää regressiovirheiden mahdollisuutta. Sovellus voi näyttää automaattisesti oikean käännöksen käyttäjän laitteen kieliasetusten mukaan, mikä parantaa käyttäjäkokemusta.

### 3.1.2 github-linkki

github.com

### 3.1.3 Kuvia ohjelman ajosta



## 3.2 Teemat

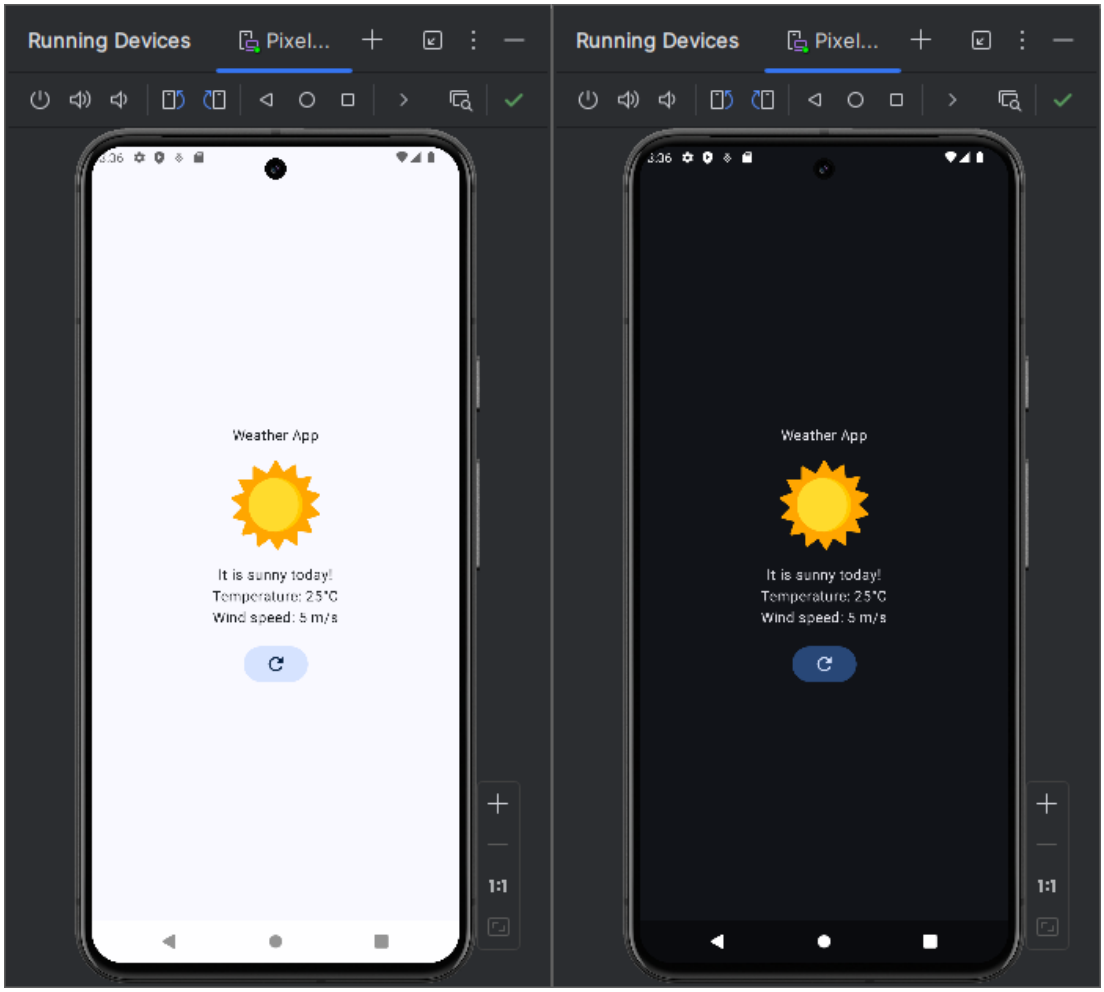
### 3.2.1 Pohdinta

Muutokset ulkoasuun voidaan tehdä helposti ja keskitetysti. Jos haluat muuttaa esimerkiksi kaikkien painikkeiden värin, voit tehdä sen teemasta käsin ilman, että sinun täytyy käydä läpi kaikkia käyttöliittymäkomponentteja erikseen. Teemat varmistavat, että koko sovelluksen ulkoasu on yhtenäinen. Väripaletti, fontit ja muut visuaaliset elementit pysyvät johdonmukaisina, mikä parantaa käyttäjäkokemusta. Teemat mahdollistavat tumman ja vaalean tilan tai jopa dynaamisten värien käytön Androidin materiaalidesignin mukaisesti. Sovellus voi mukautua käyttäjän järjestelmäasetuksiin tai antaa käyttäjälle mahdollisuuden valita eri teemoja itse.

### 3.2.2 Github-linkki

github.com

### 3.2.3 Kuvia ohjelman ajosta



## 3.3 Sovelluksen tila ja toiminnallisuus

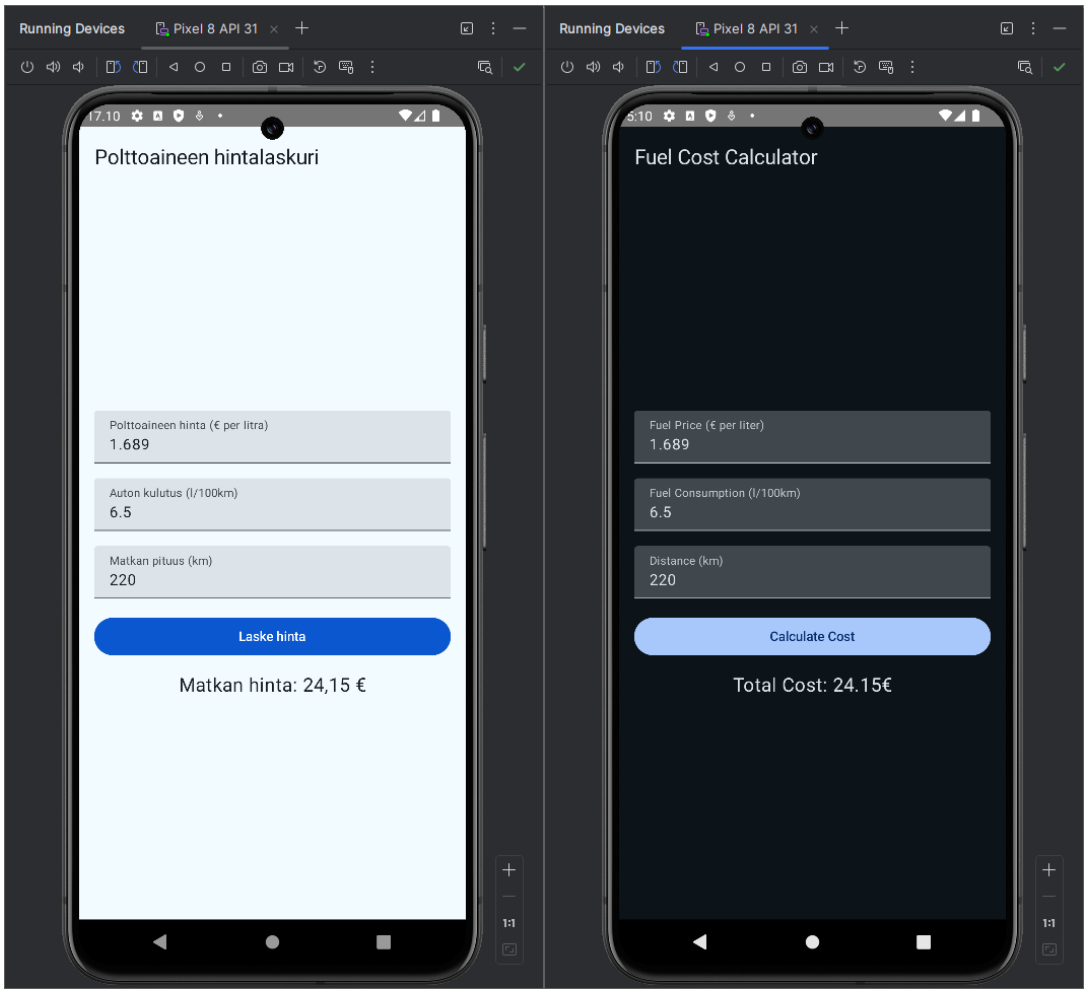
### 3.3.1 Pohdinta

Käyttöliittymän tilalla tarkoitetaan sovelluksen käyttöliittymän eri tiloja, joita voidaan käyttää käyttäjän interaktioiden seuraamiseen ja hallintaan. Näitä tiloja voi olla esimerkiksi käyttäjän syöttämät tiedot, sovelluksen näkymät tai eri elementtien tilat, kuten lomakekenttien sisällöt. Käyttöliittymän tilan hallinta on keskeistä, jotta sovelluksen eri osat voivat reagoida oikein käyttäjän tekemisiin.

### 3.3.2 Github-linkki

github.com

### 3.3.3 Kuvia ohjelman ajosta



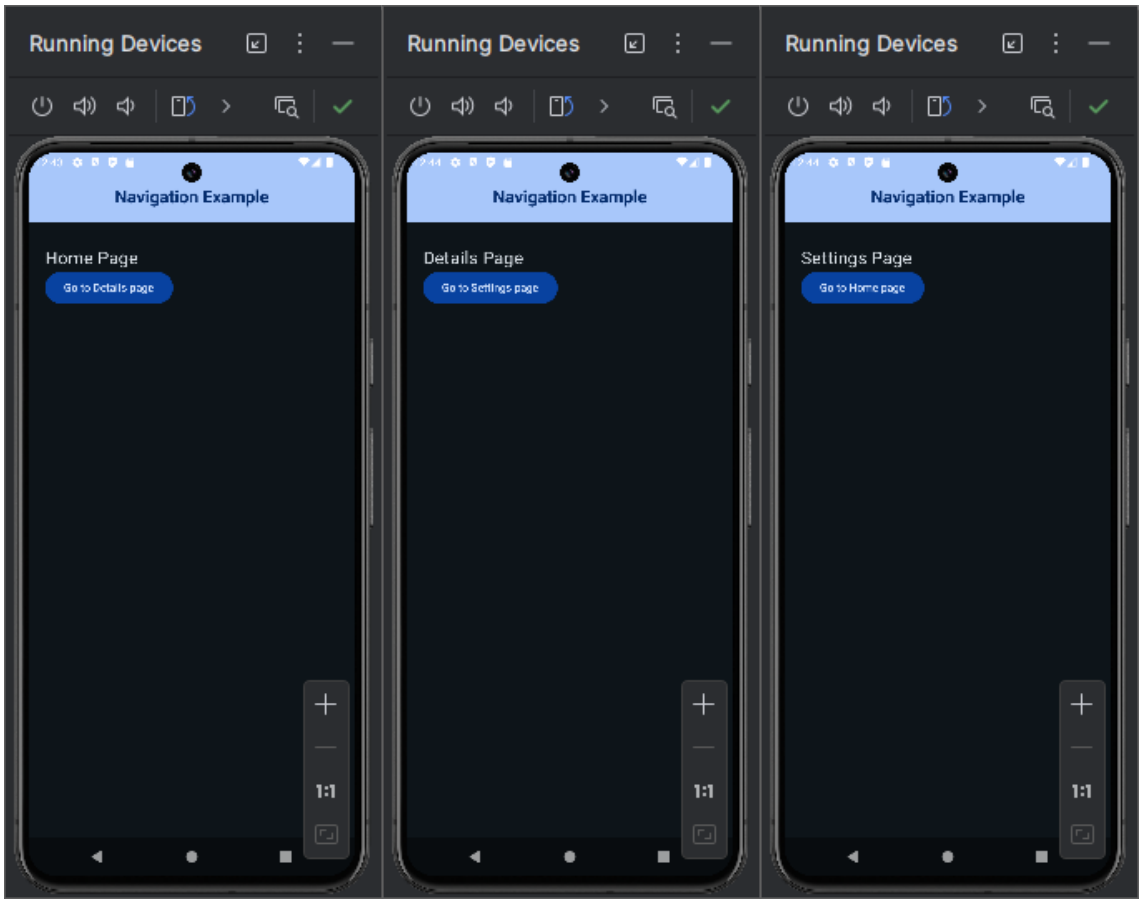
# 4 Viikkoharjoitukset 4

## 4.1 Navigointi

### 4.1.1 Github-linkki

github.com

### 4.1.2 Kuvia toiminnasta

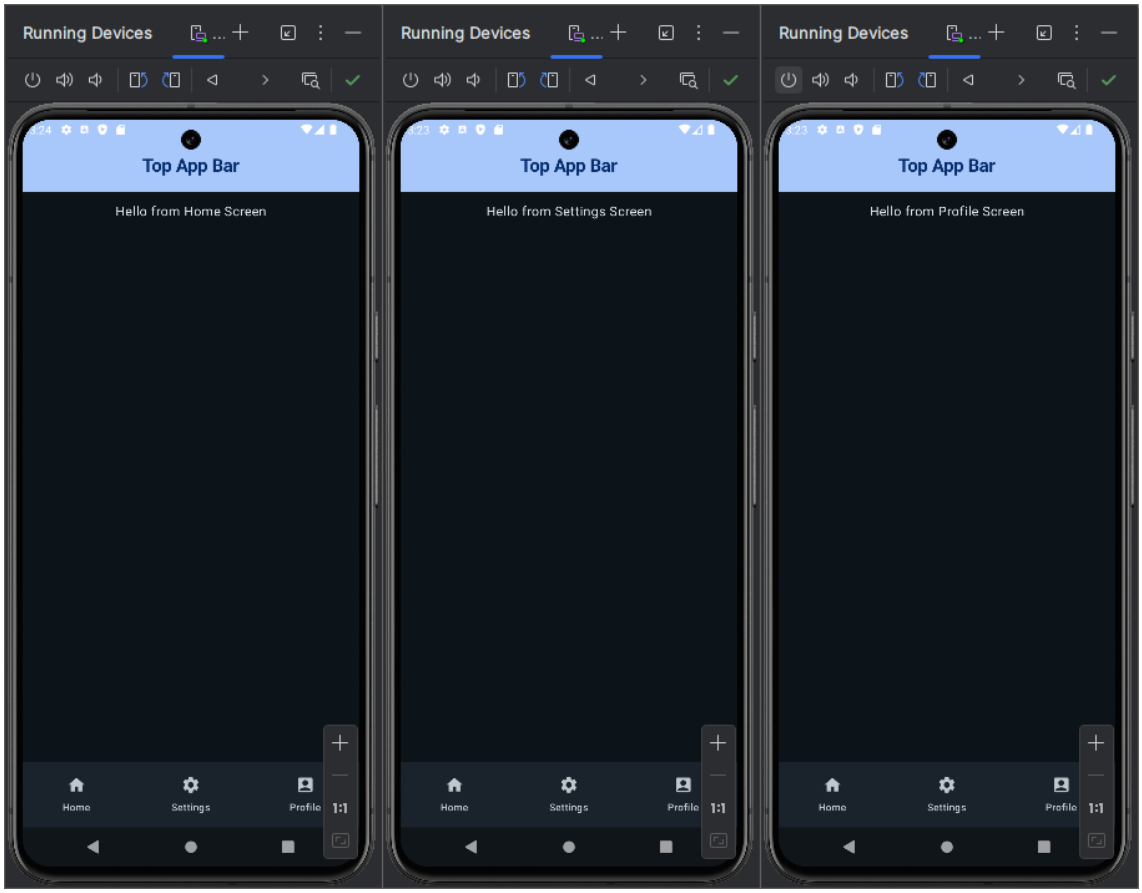


## 4.2 Bottom Tabs

### 4.2.1 Github-linkki

github.com

### 4.2.2 Kuvia toiminnasta




## 4.3 Intent

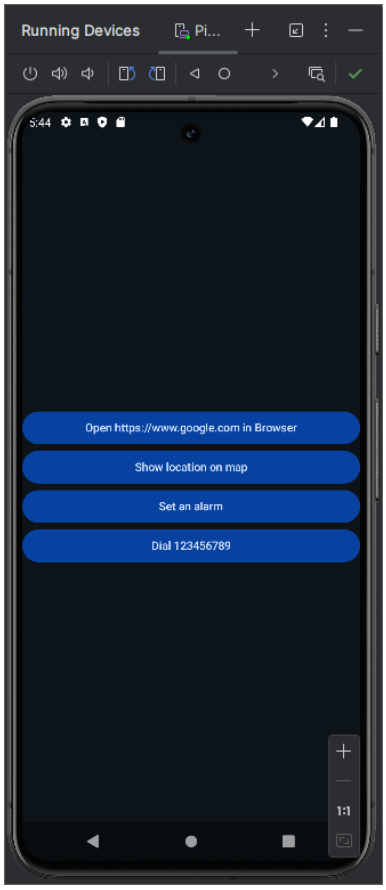
### 4.3.1 Pohdinta

Androidissa Common Intents tarkoittaa järjestelmän tarjoamia valmiita intenttejä, joita voidaan käyttää yleisiin tehtäviin sovellusten välillä. Intentti on olio, jota käytetään viestimään eri komponenttien (aktiviteettien, palveluiden jne.) välillä. Common Intents tarjoavat valmiita toimintoja, joilla sovellukset voivat käynnistää Androidin sisäänrakennettuja toimintoja tai siirtyä toisen sovelluksen tiettyyn toimintaan.

### 4.3.2 Github-linkki

github.com

### 4.3.3 Kuva ohjelmasta



# 5 Viikkoharjoitukset 5

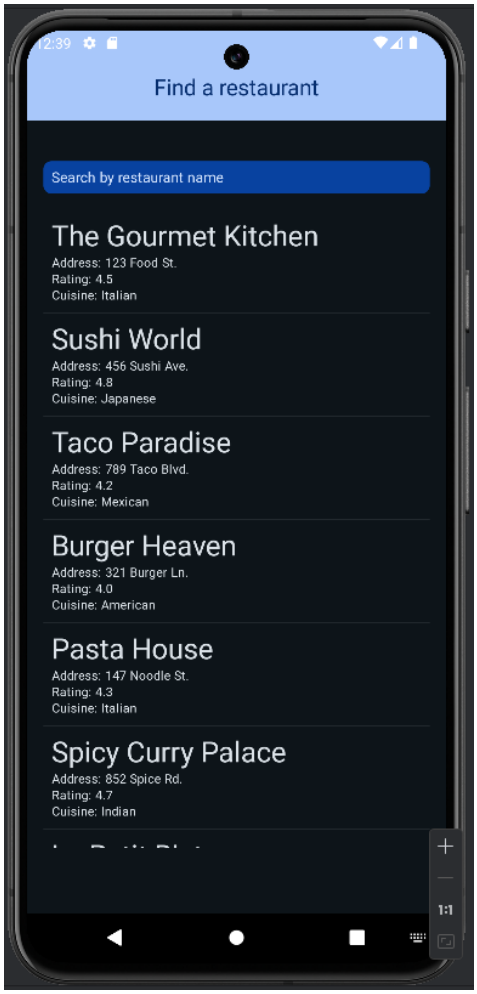
## 5.1 Dataluokat ja listojen toteuttaminen

### 5.1.1 Pohdinta

Kotlinin dataluokat (data classes) on suunniteltu tietorakenteiksi, ja ne generoivat automaattisesti tärkeät metodit kuten equals(), hashCode(), toString(), ja copy(). Javan dataluokissa nämä metodit täytyy kirjoittaa itse. Kotlinin dataluokat tukevat helppoa datan kopiointia ja toimivat sujuvasti data-binding-kirjastojen kanssa, kun taas Javassa vastaava käytettävyys vaatii lisäkoodia.

Column on yksinkertainen pystysuuntainen kontti, joka soveltuu pienille, kiinteille määrille elementtejä, sillä se renderöi kaikki elementit kerralla. LazyColumn on tarkoitettu suurille listanäkymille, ja se luo vain näkyvissä olevat elementit (lazy-loading), optimoiden suorituskyvyn ja muistinkäytön. Dynaamisissa listoissa LazyColumn on tehokkaampi vaihtoehto.

### 5.1.2 Kuva ohjelmasta



### 5.1.3 github-linkki

github.com

## 5.2 Detaljinäkymä

### 5.2.1 Kuva näkymästä


### 5.2.2 github-linkki

github.com

**Käytetyt lähteet**

https://kotlinlang.org/docs/

https://developer.android.com/guide

https://developer.android.com/studio/debug/

https://material-foundation.github.io/material-theme-builder/